Devide and Conquer - 1

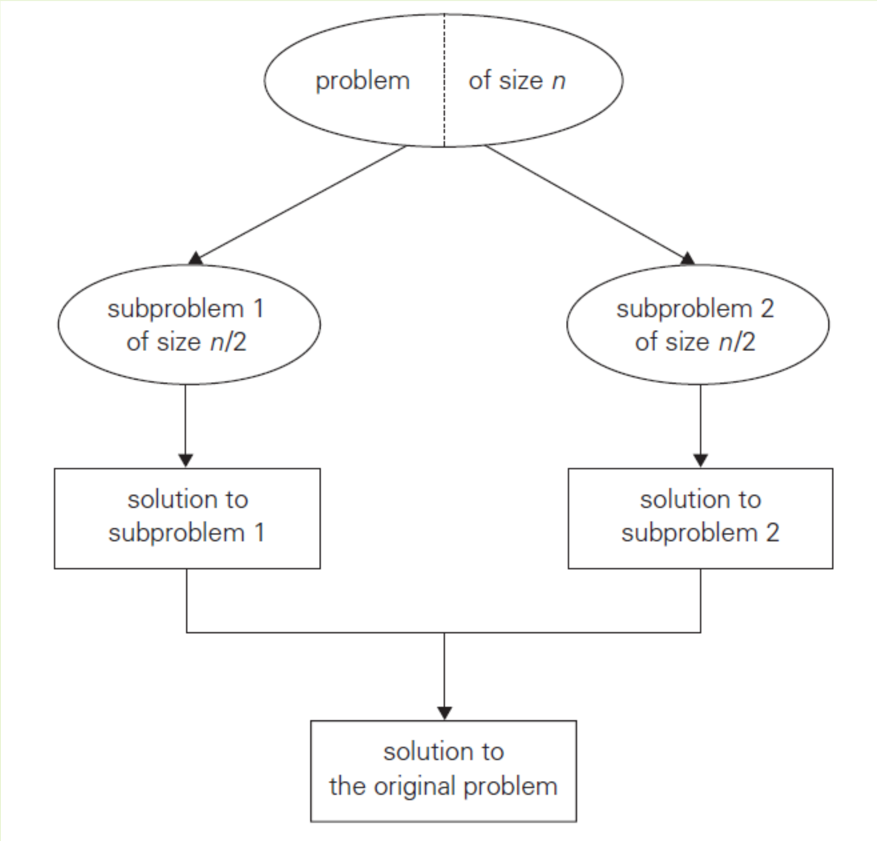
Achmad Ridok dan Tim DAA

Bahasan

- **Definisi davide and conquer**
- **Contoh-contoh kasus dengan DAC**
  1. **MinMax**
  2. **Closed pair**
  3. **Sorting problem**
     a. **Merge sort**
     b. **Quick Sort**
  4. **Strassen Matrix Multiplication**

Definisi Devide and Conquer

- **Suatu teknik algortma yang membagi penyelaian ke dalam tiga bagian berikut ini:**
  1. **Devide: Ini melibatkan pembagian masalah menjadi sub-masalah yang lebih kecil.**
  2. **Conquer: Menyelesaikan submasalah dengan memanggil secara rekursif hingga selesai.**
  3. ***Combine*: Menggabungkan sub-masalah untuk mendapatkan solusi akhir dari keseluruhan masalah.**



Skema algoritma DAC

```
DAC(a, i, j)
    if(small(a, i, j))
      return(Solution(a, i, j))
    else
      mid = divide(a, i, j)              // f1(n)
      b = DAC(a, i, mid)                 // T(n/2)
      c = DAC(a, mid+1, j)               // T(n/2)
      d = combine(b, c)                  // f2(n)
   return(d)
```

- $T(n) = \begin{cases} O(1), & if\, n\ is\ small \\ f1(n) + 2T\left(\frac{n}{2}\right) + f2(n), & \end{cases}$

Contoh-contoh kasus dengan DAC

1. MinMax
2. Closed pair
3. Sorting problem
   1. Merge sort
   2. Quick Sort
4. Binary search
5. Strassen Matrix Multiplication

1. MinMaks Problem

- **Mencari Nilai Minimum dan Maksimum (MinMaks) Persoalan:**

  Misalkan diberikan tabel *A yang berukuran n elemen dan sudah berisi nilai integer. Carilah nilai minimum dan nilai maksimum sekaligus di dalam tabel tersebut.*

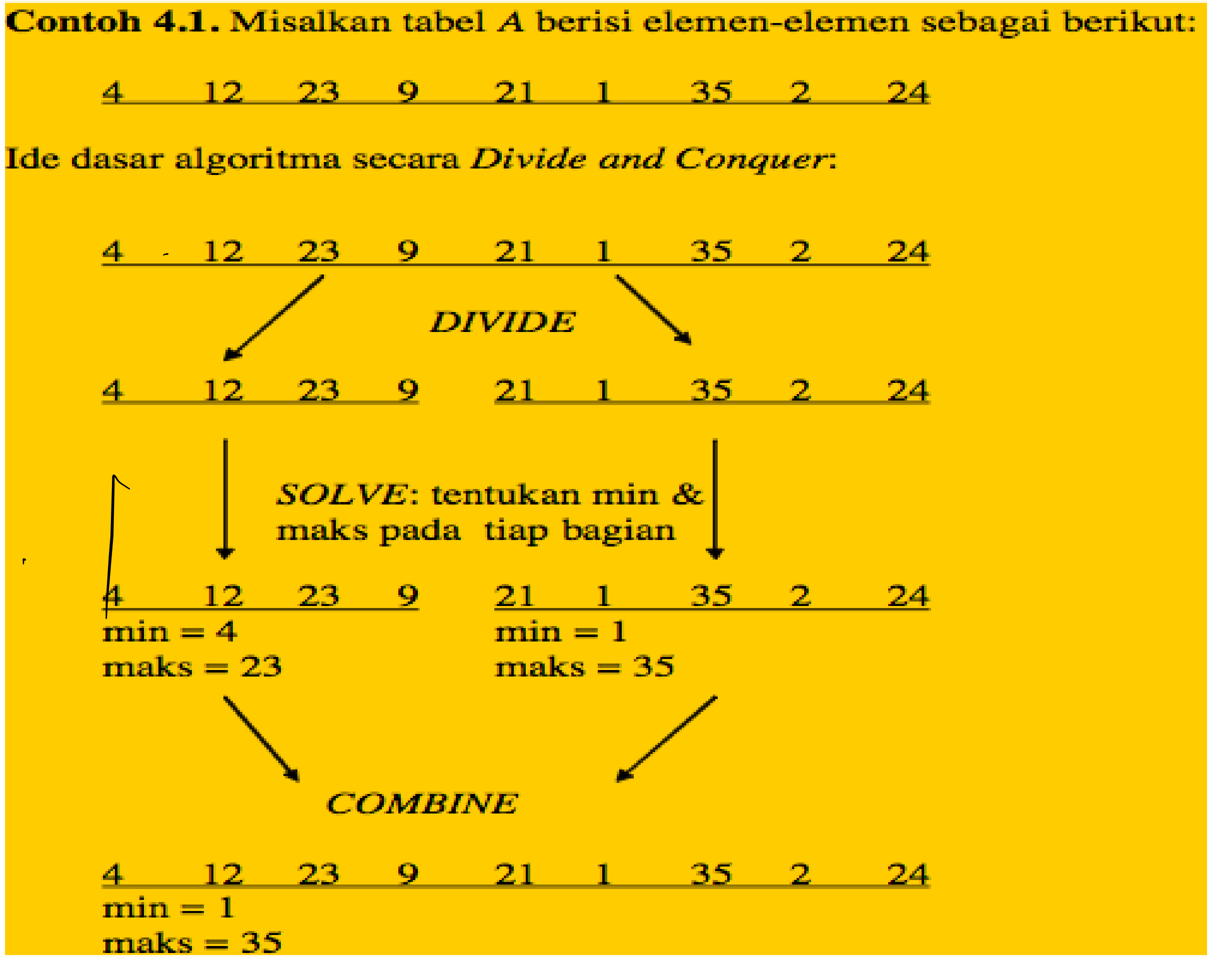
Cara Brute Force

```
MinMaks1(in A : Arr, n : integer,
              out min, maks : integer)
Deklarasi
   i : integer
Algoritma:
   min <- A[1]
   maks <- A[1]
   for i <- 2 to n do
     if Ai < min then  min <- A[i]  endif
     if Ai > maks then  maks <- A[i]  endif
  od
```

$T(n) = (n - 1) + (n - 1) = 2n - 2 = O(n)$

1. Minmax dengan DAC

- **Ukuran array hasil pembagian dapat dibuat cukup kecil sehingga mencari nilai minimum dan maksimum dapat diselesaikan (SOLVE) secara lebih mudah.**
- **Dalam hal ini, ukuran kecil yang dipilih adalah 1 elemen atau 2 elemen.**



1. Minmax dengan DAC

MinMaks(A, n, min, maks) :

- Untuk kasus n = 1 atau n = 2,
  SOLVE: Jika n = 1, maka min = maks = A[n]
  Jika n = 2, maka bandingkan kedua elemen untuk menentukan min dan maks.
- Untuk kasus n > 2,
  a) DIVIDE: Bagi array A menjadi dua bagian yang sama : A1 dan A2
  b) CONQUER: MinMaks(A1, n/2, min1, maks1)
     MinMaks(A2, n/2, min2, maks2)
  c) COMBINE:
     if min1 < min2 then min = min1
     else min = min2
     if maks1 < maks2 then maks = maks2
     else maks = maks1

DIVIDE dan CONQUER:

4 12 23 9 21 1 35 2 24

4 12 23 9 | 21 1 35 2 24

4 12 | 23 9 | 21 1 | 35 | 2 24

SOLVE dan COMBINE:

4 12	23 9	21 1	35	2 24
min = 4 maks = 12	min = 9 maks = 23	min = 1 maks = 21	min = 35 maks =35	min = 2 maks = 24

4 12 23 9	21 1	35 2 24
min = 4 maks = 23	min = 1 maks = 21	min = 2 maks = 35

4 12 23 9	21 1 35 2 24
min = 4 maks = 23	min = 1 maks = 35

4 12 23 9 21 1 5 2 24

min = 1
maks = 35

1. MinMaks dengan DAC

```
MinMaks2(in A : Arr, i, j : integer,
                   out min, maks : integer)
Algoritma:
      if i=j then                   { 1 elemen  }
        min <- A[i], maks <- A[i]
      else
        if (i = j-1) then           { 2 elemen  }
           if Ai < Aj then
                  maks <- A[j], min <- A[i]
           else
                  maks <- A[i], min <- A[j]
           fi
        else
            k <- (i+j) div 2    { lebih dari 2 elemen }
            { bagidua arr pada posisi k }
            MinMaks2(A, i, k, min1, maks1)
            MinMaks2(A, k+1, j, min2, maks2)

            if min1 < min2 then min <- min1
            else min <- min2
            fi

            if maks1 < maks2 then maks <- maks2
            else maks <- maks1
            fi
         fi
      fi
```

- **Kompleksitas Asimtotik**

$$T(n) = \begin{cases} 0 & ,n = 1 \\ 1 & ,n = 2 \\ 2T(n/2) + 2 & ,n > 2 \end{cases}$$

Penyelesaian:

Asumsi: $n = 2^k$, dengan k bilangan bulat positif, maka

$$\begin{aligned} T(n) &= 2T(n/2) + 2 \\ &= 2(2T(n/4) + 2) + 2 = 4T(n/4) + 4 + 2 \\ &= 4T(2T(n/8) + 2) + 4 + 2 = 8T(n/8) + 8 + 4 + 2 \\ &= \ldots \\ &= 2^{k-1}\,T(2) + \sum_{i=1}^{k-1} 2^i \\ &= 2^{k-1} \cdot 1 + 2^k - 2 \\ &= n/2 + n - 2 \\ &= 3n/2 - 2 \\ &= O(n) \end{aligned}$$

1. MinMax - Analisis

MinMaks1 secara *brute force* :

$T(n) = 2n - 2$

MinMaks2 *secara divide and conquer:*

$T(n) = 3n/2 - 2$

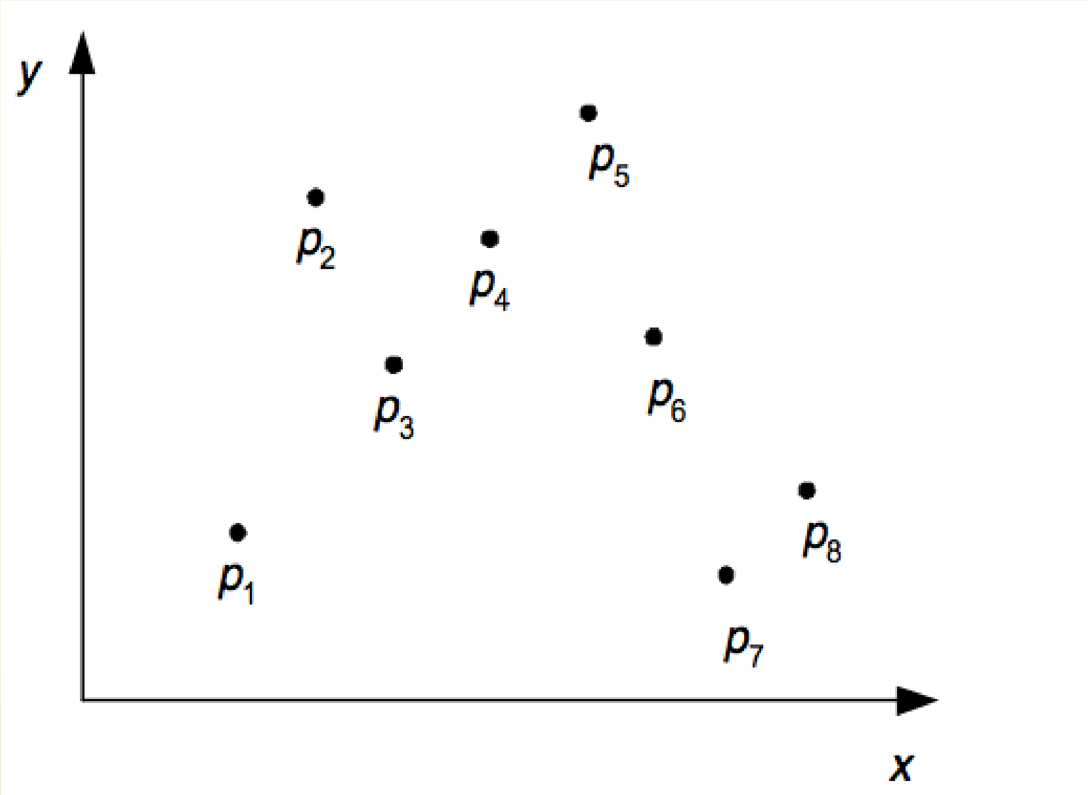
Perhatikan: $3n/2 - 2 < 2n - 2$, $n \geq 2$.

Kesimpulan: algoritma MinMaks lebih efektif dengan metode Divide and Conquer.

2. Closest pair Problem

- **Persoalan:**

  Diberikan himpunan titik, *P, yang terdiri dari n buah titik, (xi, yi), pada bidang 2-D. Tentukan jarak terdekat antara dua buah titik di dalam himpunan P.*



2. Close Pair dengan algoritma Brute Force

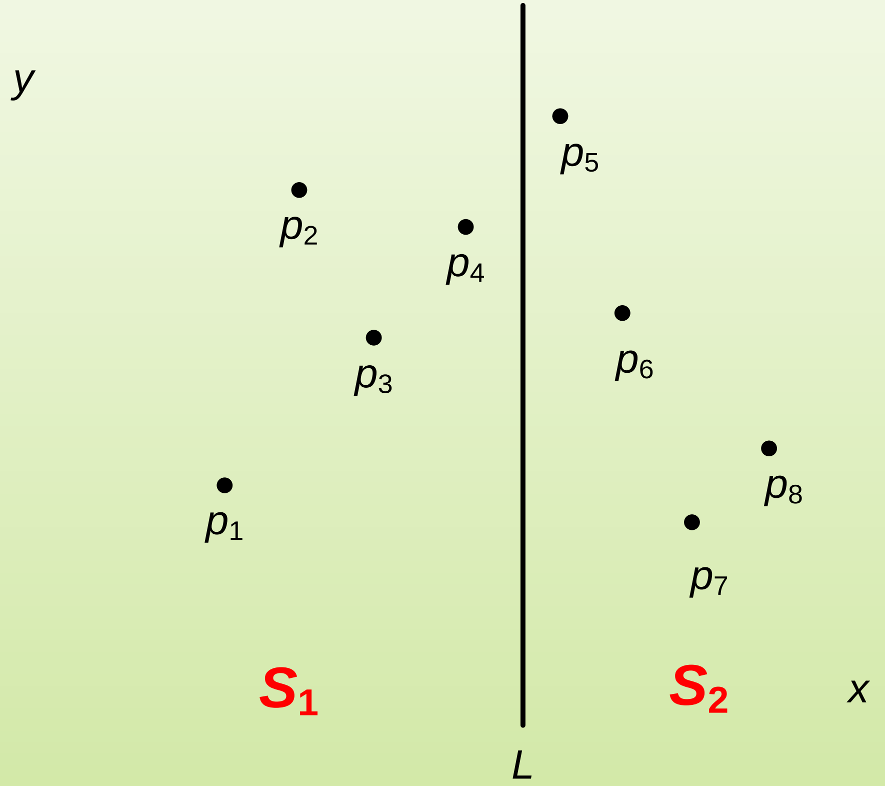
- **Hitung jarak setiap pasang titik. Ada sebanyak *$C(n, 2) = n(n - 1)/2$ pasangan titik***
- ***Pilih pasangan titik yang mempunyai jarak terkecil.***
- **Kompleksitas algoritma adalah *$O(n^2)$.***

2. Close Pair dengan DAC

- **Asumsi: *n* = 2^k dan titik-titik sudah diurut berdasarkan absis (*x*).**

Algoritma *Closest Pair*:

1. **SOLVE: jika *n* = 2, maka jarak kedua titik dihitung langsung dengan rumus Euclidean.**
2. **DEVIDE : Bagi himpunan titik ke dalam dua bagian, S_1 dan S_2, setiap bagian mempunyai jumlah titik yang sama. *L* adalah garis maya yang membagi dua himpunan titik ke dalam dua sub-himpunan, masing-masin *n*/2 titik.**

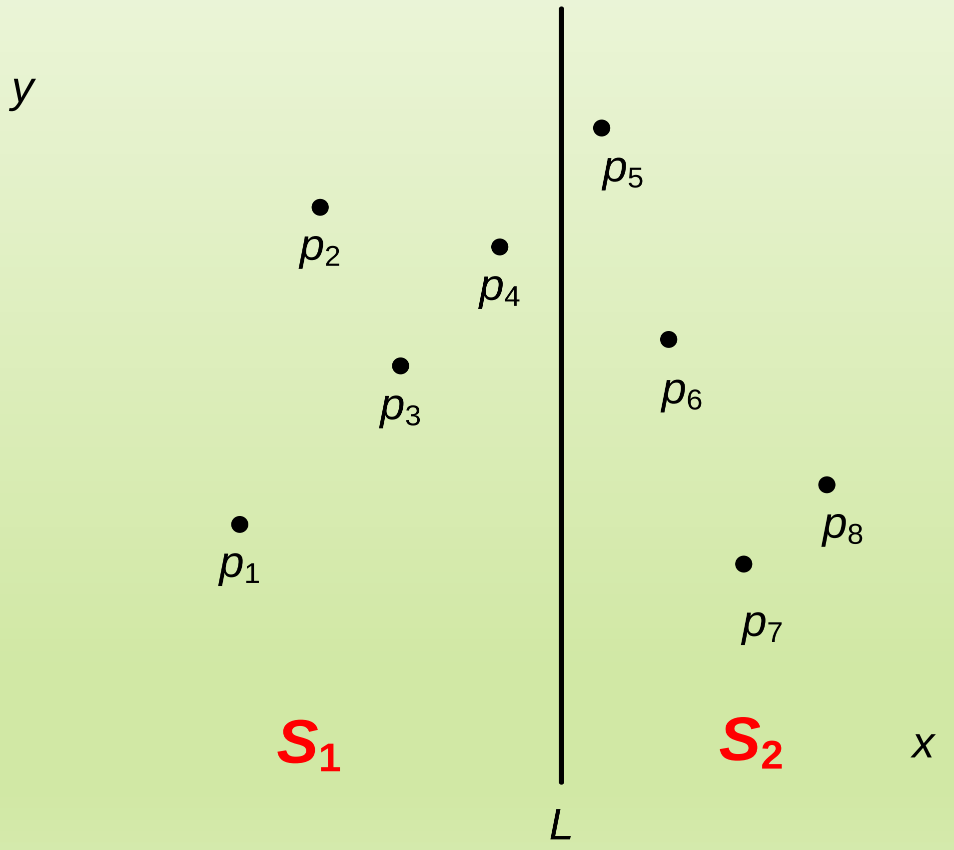


Garis *L* dapat dihampiri sebagai $y = x_{n/2}$ dengan asumsi titik-titik diurut menaik berdasarkan absis.

2. Close Pair dengan DAC

3. **CONQUER: Secara rekursif, terapkan algoritma *D-and-C* pada masing-masing bagian.**
4. **COMBINE: Pasangan titik yang jaraknya terdekat ada tiga kemungkinan letaknya:**
   a) **Pasangan titik terdekat terdapat di bagian S_1.**
   b) **Pasangan titik terdekat terdapat di bagian S_2.**
   c) **Pasangan titik terdekat dipisahkan oleh garis batas *L*, yaitu satu titik di S_1 dan satu titik di S_2.**

   Jika kasusnya adalah (c), maka lakukan tahap ketiga untuk mendapatkan jarak dua titik terdekat sebagai solusi persoalan semula.

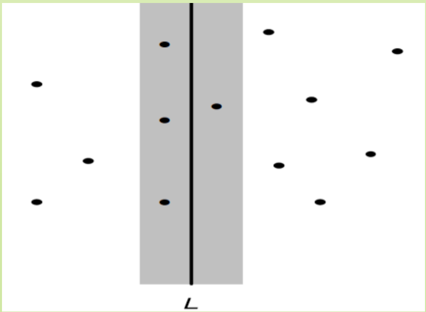


2. Close Pair dengan DAC

```
FindClosestPair2(input P: SetOfPoint, n : integer,
                 output d : real)
Deklarasi:
  d1, d2 : real

Algoritma:
  if n = 2 then
    d <- jarak kedua titik dengan rumus Euclidean
  else
    S1 <- {p1, p2 ,..., pn/2 }
    S2 <- {pn/2+1, pn/2+2 ,..., pn }
    FindClosestPair2(S1, n/2, d1)
    FindClosestPair2(S2, n/2, d2)
    d <- MIN(d1,d2)
    {--********************************************--}
    Tentukan apakah terdapat titik pl di S1 dan pr di
    S2 dengan jarak(pl, pr) < d. Jika ada, set
    d dengan jarak terkecil tersebut.
    {--********************************************--}
  fi
```

- **Jika terdapat pasangan titik p_l and p_r yang jaraknya lebih kecil dari *d*, maka kasusnya adalah:**
  - **Absis *x* dari p_l dan p_r berbeda paling banyak sebesar d**
  - **Ordinat *y* dari p_l dan p_r berbeda paling banyak sebesar *d*.**
- **Ini berarti p_l and p_r adalah sepasang titik yang berada di daerah sekitar garis vertikal *L*:**



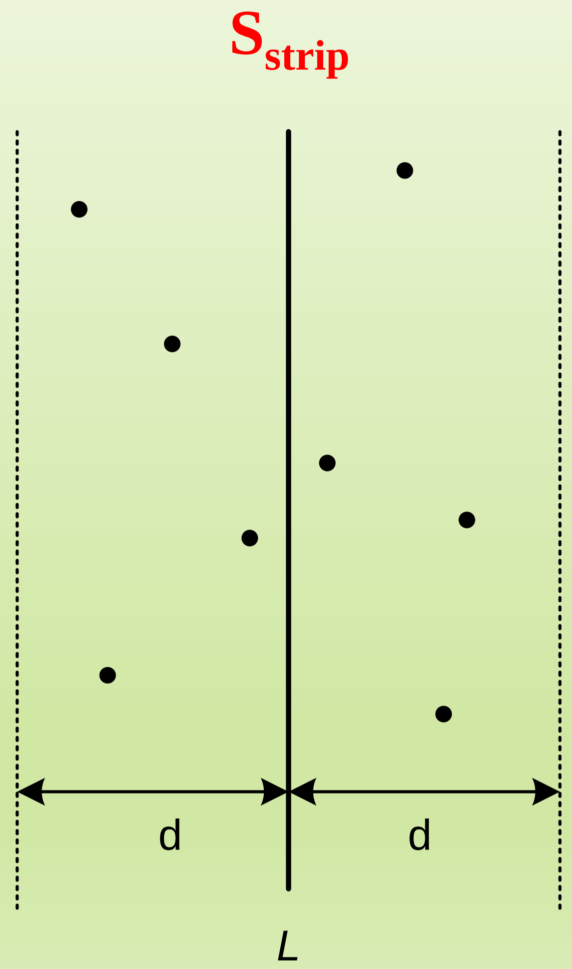
- **Berapa lebar strip abu-abu tersebut?**

2. Close Pair dengan DAC

- **Kita membatasi titik-titik di dalam *strip* selebar 2*d***
- **Oleh karena itu, implementasi tahap ketiga adalah sbb:**
  i. **Temukan semua titik di $S1_t$ yang memiliki absis x minimal $x_{n/2}$ – d.**
  ii. **Temukan semua titik di *S2* yang memiliki absis *x* maksimal $x_{n/2} + d$.**

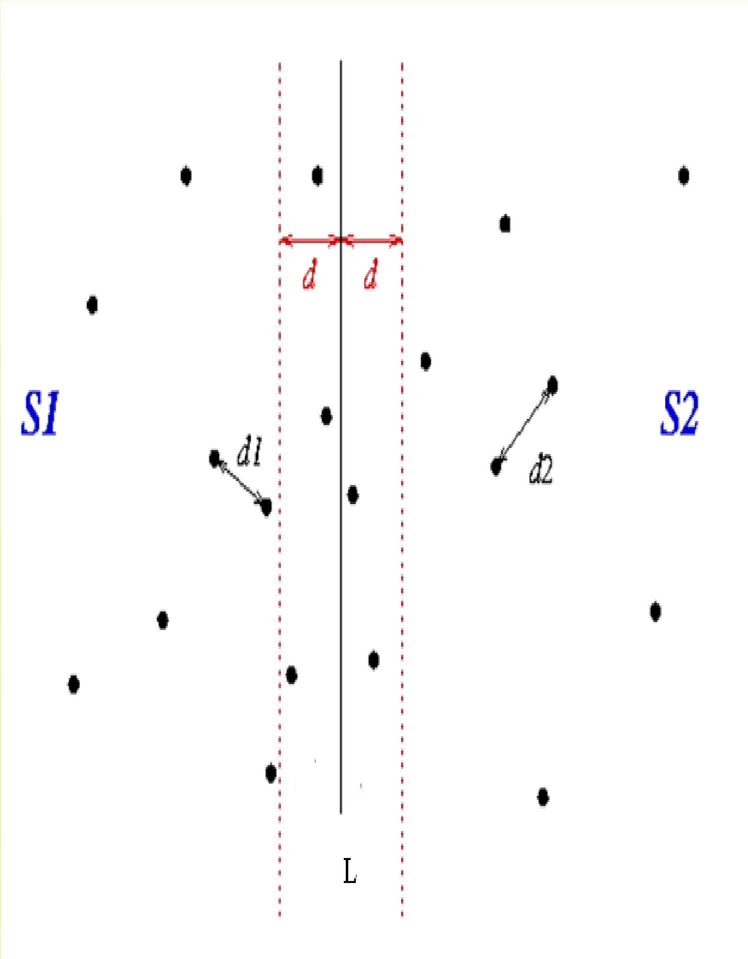
Sebut semua titik-titik yang ditemukan pada langkah (i) dan (ii) tersebut sebagai himpunan S_{strip} yang berisi *s* buah titik.

Urutkan titik-titik tersebut dalam urutan ordinat *y* yang menaik. Misalkan *q*1, *q*2 , ..., *qs* menyatakan hasil pengurutan.



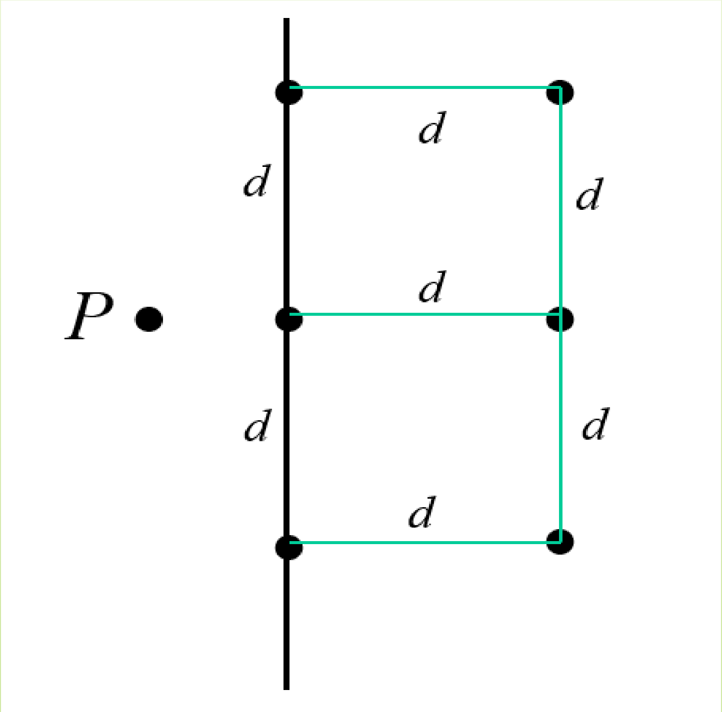
2. Close Pair dengan DAC

```
for i <- 1 to s do
   for j <- i+1 to s do
     if (|qi.x-qj.x| > d or |qi.y-qj.y| > d then
        tidak diproses
     else
        d3 <- EUCLIDEAN(qi, qj)
        if d3 < d then d <- d3 fi
     fi
   od
  od
```



2. Close Pair dengan DAC

- **Jika diamati, kita tidak perlu memeriksa semua titik di dalam area strip abu-abu tersebut.**
- **Untuk sebuah titik *P* di sebelah kiri garis L, kita hanya perlu memeriksa paling banyak enam buah titik saja yang jaraknya sebesar *d* dari ordinat *P* (ke atas dan ke bawah), serta titik-titik yang berjarak *d* dari garis *L.***



Untuk memahami konsep ini silakan simak video ini :
https://youtu.be/6u_hWxbOc7E

2. Close Pair dengan DAC

- **Pengurutan titik-titik dalam absis *x* dan ordinat *y* dilakukan sebelum menerapkan algoritma *Divide and Conquer*.**
- **Pemrosesan titik-titk di dalam S_{strip} butuh waktu $t(n) = cn = O(n)$.**
- **Kompleksitas algoritma:**

$$T(n) = \begin{cases} 2T(n/2) + cn & , n > 2 \\ a & , n = 2 \end{cases}$$

- **Solusi dari persamaan di atas adalah $T(n) = O(n \log n)$, sesuai dengan Teorema Master**

3. Sorting problem

```
procedure Sort(input/output A : TabelInt, input n : integer)

{ Mengurutkan tabel A dengan metode Divide and Conquer
  Masukan: Tabel A dengan n elemen
  Keluaran: Tabel A yang terurut
}
Algoritma:
   if Ukuran(A) > 1 then
      Bagi A menjadi dua bagian, A1 dan A2, masing-masing berukuran n1
      dan n2 (n = n1 + n2)

      Sort(A1, n1)   { urut bagian kiri yang berukuran n1 elemen }
      Sort(A2, n2)   { urut bagian kanan yang berukuran n2 elemen }

      Combine(A1, A2, A)  { gabung hasil pengurutan bagian kiri dan
                            bagian kanan }
   end
```

3. Sorting problem dengan DAC

A	4	12	3	9	1	21	5	2

Dua pendekatan pengurutan:

1.Mudah membagi, sulit menggabung (easy split/hard join)

Tabel A dibagi dua berdasarkan posisi elemen:

Divide :

A1	4	12	3	9	A2	1	21	5	2

Sort:

A1	3	4	9	12	A2	1	2	5	21

Combine:

A1	1	2	3	4	5	9	12	21

Algoritma pengurutan yang termasuk jenis ini: Merge Sort

3. Sorting problem dengan DAC

A	4	12	3	9	1	21	5	2

2. Sulit membagi, mudah menggabung *(hard split/easy join)*

Tabel A dibagi dua berdasarkan nilai elemennya. Misalkan elemen-elemen A1 ≤ elemen-elemen A2.

Divide:

A1	4	2	3	1	A2	9	21	5	12

Sort:

A1	1	2	3	4	A2	5	9	12	21

Combine:

A	1	2	3	4	5	9	12	21

Algoritma pengurutan yang termasuk jenis ini: *Quick Sort*

3.a. Merge sort

Untuk kasus n = 1, maka tabel A sudah terurut dengan sendirinya (langkah SOLVE). Untuk kasus n > 1, maka

a. DIVIDE: bagi tabel A menjadi dua bagian, bagian kiri dan bagian kanan, masing-masing bagian berukuran n/2 elemen.

b. CONQUER: Secara rekursif, terapkan algoritma D-and-C pada masing-masing bagian.

c. MERGE: gabung hasil pengurutan kedua bagian sehingga diperoleh tabel A yang terurut.

3.a. Merge sort - Contoh

A1			A2				B					
1	13	24	2	15	27	1 < 2 → 1	1					
1	13	24	2	15	27	2 <13 → 2	1	2				
1	13	24	2	15	27	13<15→13	1	2	13			
1	13	24	2	15	27	15<24→15	1	2	13	15		
1	13	24	2	15	27	24<27→24	1	2	13	15	24	
1	13	24	2	15	27	27 →	1	2	13	15	24	27

3.a. Merge sort - Contoh

4 12 23 9 21 1 5 2

DIVIDE, CONQUER, dan SOLVE:

4 12 23 9 21 1 5 2

4 12 23 9 21 1 5 2

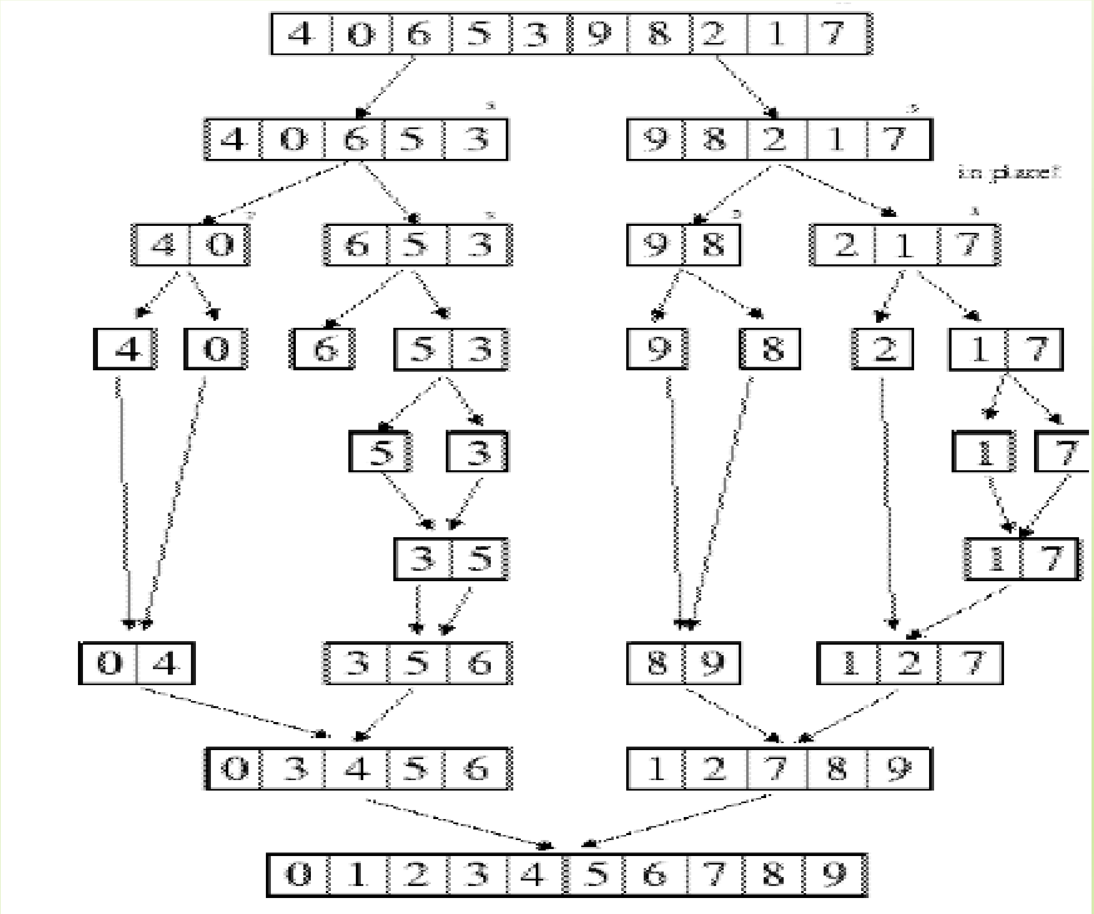
4 12 23 9 21 1 5 2

4 12 23 9 21 1 5 2

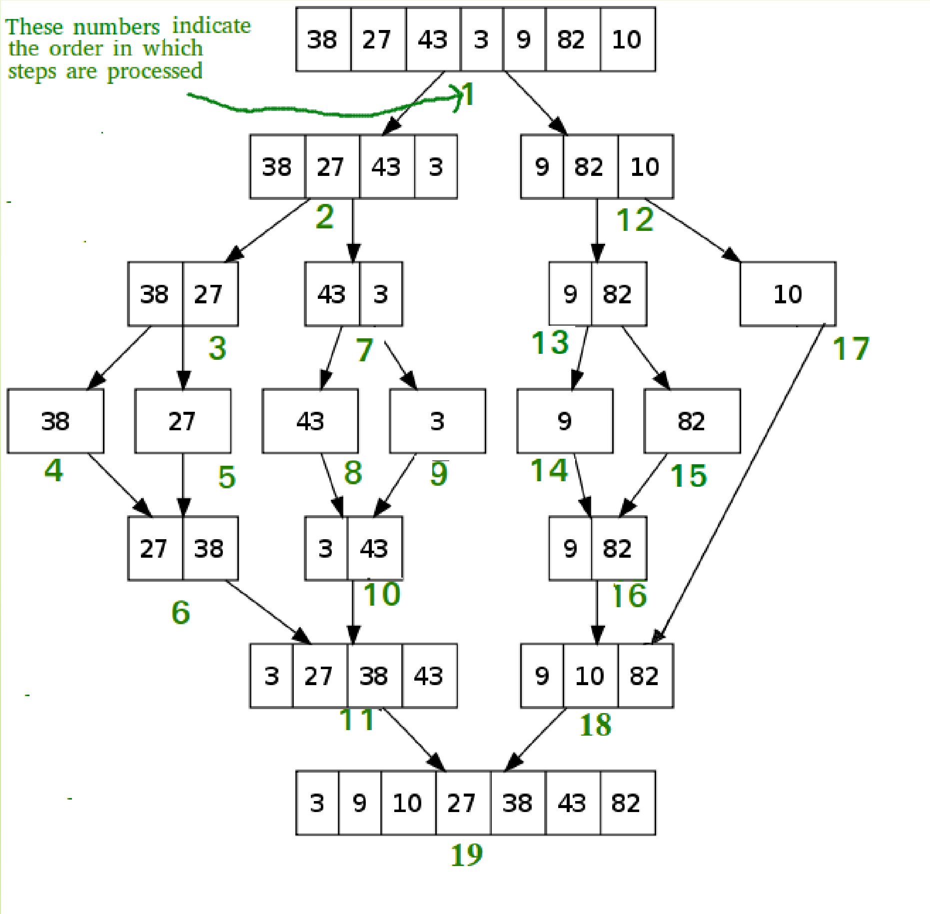
MERGE: 4 12 9 23 1 21 2 5

4 9 12 23 1 2 5 21

1 2 4 5 9 12 21 23



3.a. Merge sort - Contoh



mergeSort(arr[], l, r)

If r > l

middle m = l + (r – l)/2

mergeSort(arr, l, m)

mergeSort(arr, m + 1, r)

merge(arr, l, m, r)

3.a. Merge sort – Merge

```
procedure Merge(in/out A:Arr,
                   input kiri,tengah,kanan: integer)
Deklarasi
          B : Arr
          i, id1, id2 : integer

Algoritma:
          id1 <- ki,  id2 <- tengah + 1, i <- kiri
          while (id1 <= tengah) and (id2 <= kanan) do
          if A[id1] <- A[id2] then
                    Bi <- A[id1]
                              id1 <- id1 + 1
                    else
                    Bi <- id2
                              id2 <- id2 + 1
                    endif
                    i <- i + 1
          od
while (id1 <= tengah) do
          Bi <- A[id1]
          id1 <- id1 + 1
          i <- i + 1
od
while (id2 <=  kanan) do
          Bi <- A[id2]
          id2 <- id2 + 1
          i <- i + 1
od
for i <- kiri to kanan do
          A[i] <- B[i]
od
```

3.a. Merge sort - Contoh

- **Kompleksitas waktu:**
  - **Asumsi:** $n = 2^k$
  - ***T(n) = jumlah perbandingan pada pengurutan dua buah sub-tabel + jumlah perbandingan pada prosedur Merge***

$$T(n) = \begin{cases} a & ,n = 1 \\ 2T(n/2) + cn & ,n > 1 \end{cases}$$

Penyelesaian:

$$\begin{aligned} T(n) &= 2T(n/2) + cn \\ &= 2(2T(n/4) + cn/2) + cn \\ &= 4T(n/4) + 2cn \\ &= 4(2T(n/8) + cn/4) + 2cn \\ &= 8T(n/8) + 3cn = \ldots = 2^k\, T(n/2^k) + kcn \end{aligned}$$

- ***Berhenti jika ukuran tabel terkecil,***

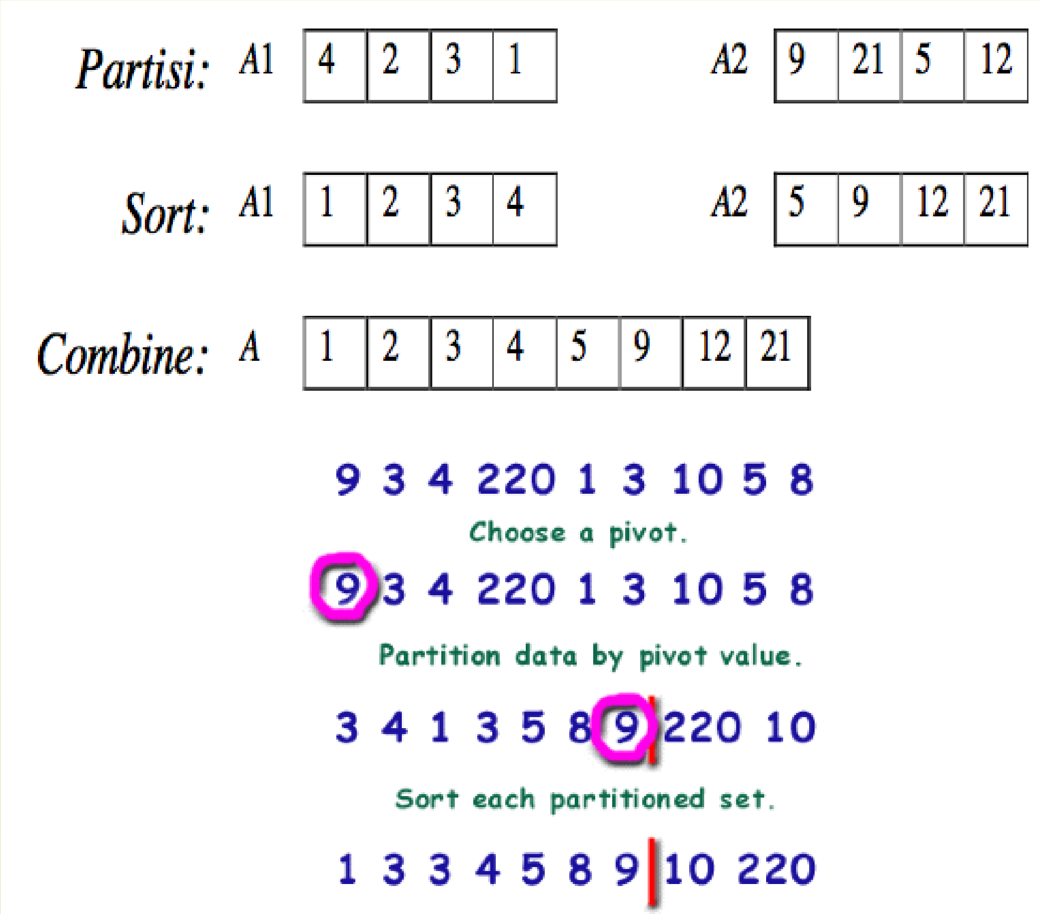
  $n = 1$: $n/2^k = 1 \rightarrow k = {}^2\log n$

- ***Sehingga***

$$\begin{aligned} T(n) &= nT(1) + cn\ {}^2\log n \\ &= na + cn\ {}^2\log n \\ &= O(n\ {}^2\log n) \end{aligned}$$

3.b. Quick sort

- Termasuk pada pendekatan sulit membagi, mudah menggabung (hard split/easy join)
- Tabel A dibagi A1 dan A2 sedemikian sehingga elemen- elemen A1 ≤ elemen-elemen A2.
- **QuickSort** mengambil elemen sebagai pivot dan mempartisi array yang diberikan di sekitar pivot yang dipilih.
- Ada beberapa versi quicksort :
  - pilih elemen pertama sebagai pivot.
  - pilih elemen terakhir sebagai pivot
  - Pilih elemen acak sebagai pivot.
  - Pilih median sebagai pivot.



3.b. Quick sort - Algoritma

```
QuickSort(in/out A : Arr, in i,j: integer)

Deklarasi
  k : integer

Algoritma:
   if  i < j then
        Partisi(A, i, j, k)
        QuickSort(A, i, k)
        QuickSort(A, k+1, j)
   endif
```

```
procedure Partisi(in/out A : Arr, in i, j : integer, out q : integer)
Deklarasi
  pivot, temp : integer
Algoritma:
  pivot <- A[(i + j) div 2],  p <- i, q <- j
  repeat
    while A[p] < pivot do
       p <- p + 1
    od
    while A[q] > pivot do
       q <- q – 1
    od
    if p < q  then
      swap(A[p], A[q])
      p <- p + 1
      q <- q - 1
    fi
  until p > q
```

3.b. Quick sort - Partisi

```
procedure Partisi(in/out A : Arr, in i, j : integer, out q
: integer)
Deklarasi
  pivot, temp : integer
Algoritma:
  pivot <- A[(i + j) div 2],  p <- i, q <- j
  repeat
    while A[p] < pivot do  p <- p + 1  od
    while A[q] > pivot do  q <- q – 1  od
    if p < q  then
      swap(A[p], A[q])
      p <- p + 1
      q <- q - 1
    fi
  until p > q
```

Teknik mempartisi tabel :

i. pilih x ∈ { A[1], A[2], ..., A[n] } sebagai pivot

ii. pindai tabel dari kiri sampai ditemukan A[p] ≥ x

iii. pindai tabel dari kanan sampai ditemukan A[q] ≤ x

iv. pertukarkan A[p] ⇔ A[q]

v. ulangi (ii), dari posisi p + 1, dan (iii), dari posisi q – 1, sampai kedua pemindaian bertemu di tengah tabel

3.b. Quick sort - Partisi

- **Misal suatu tabel berisikan elemen-elemen sebagai berikut :**

8	1	4	6	9	3	5	7

- **Langkah-langkah melakukan partisi :**
- **(i)**

8	1	4	$\boxed{6}$	9	3	5	7
			pivot				

- **(ii)&(iii)**

→							←
8	1	4	6	9	3	5	7
↑p						↑q	

- **(iv)**

5	1	4	6	9	3	8	7

- **(ii)&(iii)**

	→				←		
5	1	4	6	9	3	8	7
			↑p		↑q		

- **(iv)**

5	1	4	3	9	6	8	7

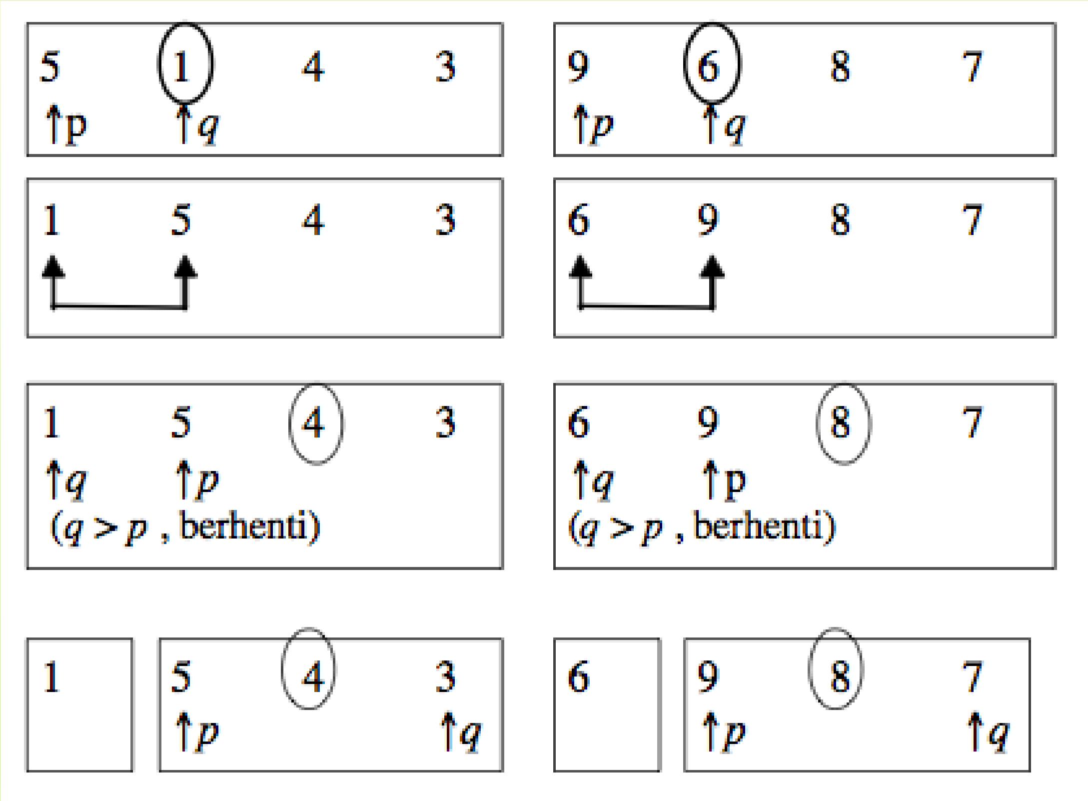
- **(ii)&(iii)**

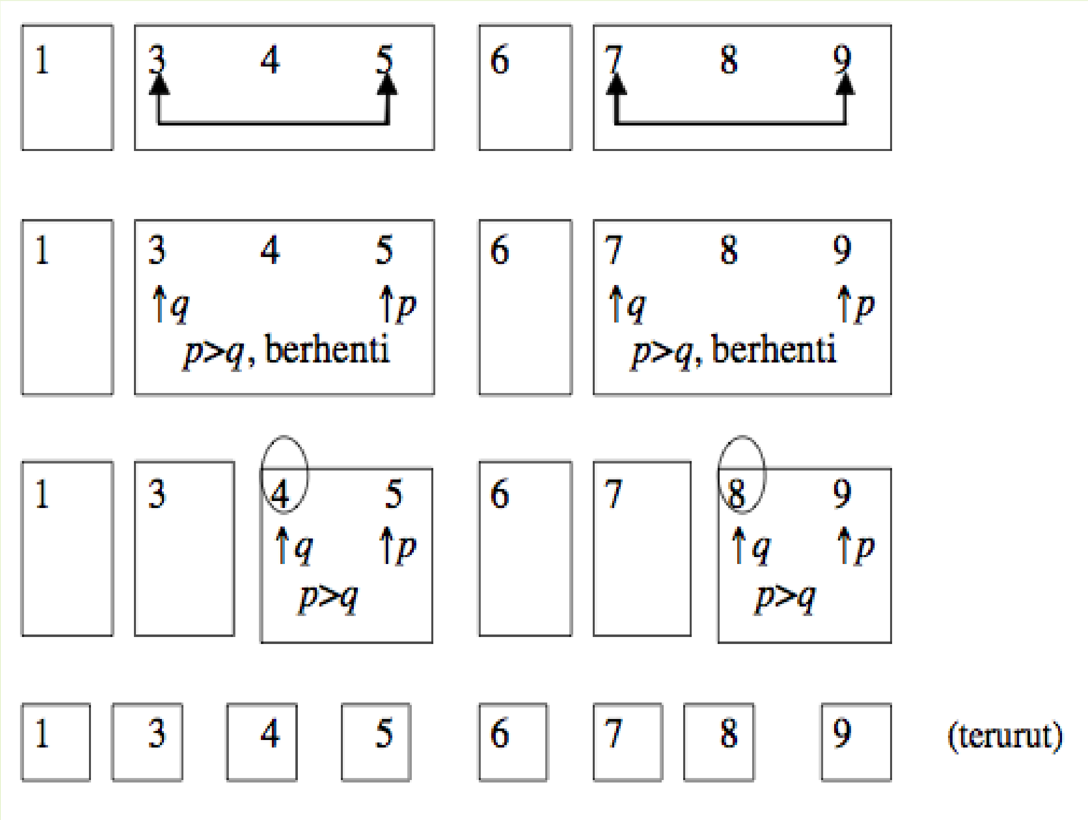
				→←			
5	1	4	3	9	6	8	7
			↑q	↑p	($q < p$, berhenti)		

- **Hasil partisi pertama :**

kiri:	5	1	4	3	(< 6)
kanan:	9	6	8	7	(≥ 6)

3.b. Quick sort - Partisi





3.b. Quick sort - Pivot

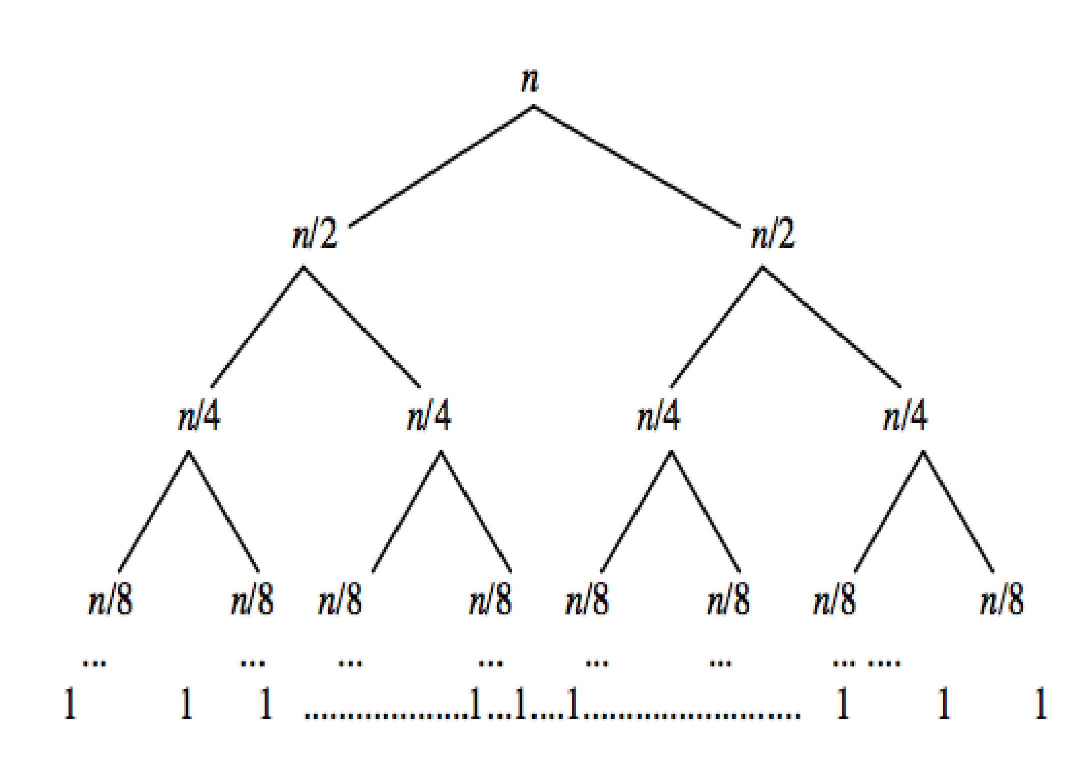
Cara pemilihan pivot:

1. **Pivot = elemen pertama/elemen terakhir/elemen tengah tabel**
2. **Pivot dipilih secara acak dari salah satu elemen tabel.**
3. **Pivot = elemen median tabel**

3.b. Quick sort - Kasus terbaik (best case)

- **Kasus terbaik terjadi bila pivot adalah elemen median sedemikian sehingga kedua sub-tabel berukuran relatif sama setiap kali pempartisian.**

$$T(n) = \begin{cases} a & ,n = 1 \\ 2T(n/2) + cn & ,n > 1 \end{cases}$$



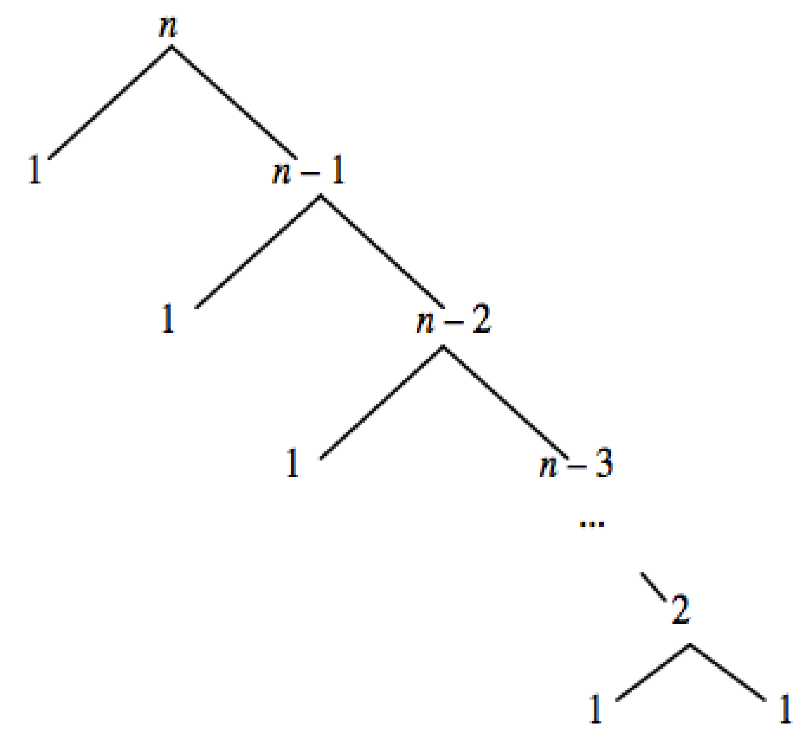
$$T(n) = 2T(n/2) + cn = na + \text{cn}\ {}^2\log n = O(n\ {}^2\log n)$$

3.b. Quick sort - Kasus terburuk (worst case)

- **Kasus ini terjadi bila pada setiap partisi pivot selalu elemen maksimum (atau elemen minimum) tabel.**
- **Kasus jika tabel sudah terurut menaik/menurun**

$$T(n) = \begin{cases} a & ,n = 1 \\ T(n-1) + cn & ,n > 1 \end{cases}$$

- **T(n) = ? (Selesaikan)**



3.b. Quick sort - *Kasus rata-rata (average case)*

- **Kasus ini terjadi jika pivot dipilih secara acak dari elemen tabel, dan peluang setiap elemen dipilih menjadi pivot adalah sama.**
- **Tavg(n) = O(n ^{2}log n)**

Teorema Master

Misalkan $T(n)$ adalah fungsi menaik yang memenuhi relasi rekurens:

$$T(n) = aT(n/b) + cn^d$$

yang dalam hal ini $n = b^k$, k = 1, 2, …, $a \geq 1$, $b \geq 2$, dan c dan d adalah bilangan riil ≥ 0, maka

$T(n)$ adalah
$$\begin{cases} O(n^d) & \text{jika } a < b^d \\ O(n^d \log n) & \text{jika } a = b^d \\ O(n^{\log_b a}) & \text{jika } a > b^d \end{cases}$$

Contoh: Pada algoritma Mergesort/Quick Sort,

$$T(n) = \begin{cases} a & , n = 1 \\ 2T(n/2) + cn & , n > 1 \end{cases}$$

Menurut Teorema Master, a = 2, b = 2, d = 1, dan $a = b^d$, maka relasi rekurens:

$$T(n) = 2T(n/2) + cn = O(n \log n)$$

4. Perkalian Matriks

- **Misalkan *A* dan *B* dua buah matrik berukuran $n \times n$.**
- **Perkalian matriks: $C = A \times B$**

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj} = \sum_{k=1}^{n} a_{ik}b_{kj}$$

Metode Brute Force

```
function KaliMatriks1(input A,B: Matriks,
                      input n : integer) : Matriks
Algoritma:
  for i <- 1 to n do
    for j <- 1 to n do
      C[i,j] <- 0
      for k <- 1 to n do
        C[i,j] <- C[i,j] + A[i,k] * B[k,j]
      od
    od
  od
  return C
```

Kompleksitas algoritma: $T(n) = n^3 + n^3 = O(n^3)$.

4. Perkalian Matriks - DAC

Matriks A dan B dibagi menjadi 4 buah matriks bujur sangkar. Masing-masing matriks bujur sangkar berukuran $n/2 \times n/2$:

$$\underset{\mathbf{A}}{\begin{bmatrix} A11 & A12 \\ A21 & A22 \end{bmatrix}} \times \underset{\mathbf{B}}{\begin{bmatrix} B11 & B12 \\ B21 & B22 \end{bmatrix}} = \underset{\mathbf{C}}{\begin{bmatrix} C11 & C12 \\ C21 & C22 \end{bmatrix}}$$

Elemen-elemen matriks C adalah:

$$C11 = A11 \cdot B11 + A12 \cdot B21$$
$$C12 = A11 \cdot B12 + A12 \cdot B22$$
$$C21 = A21 \cdot B11 + A22 \cdot B21$$
$$C22 = A21 \cdot B12 + A22 \cdot B22$$

4. Perkalian Matriks - Contoh

Misalkan matriks A adalah sebagai berikut:

$$A = \begin{bmatrix} 3 & 4 & 8 & 16 \\ 21 & 5 & 12 & 10 \\ 5 & 1 & 2 & 3 \\ 45 & 9 & 0 & -1 \end{bmatrix}$$

Matriks A dibagi menjadi 4 upa-matriks 2 x 2:

$$A11 = \begin{bmatrix} 3 & 4 \\ 21 & 5 \end{bmatrix} \quad A12 = \begin{bmatrix} 8 & 16 \\ 12 & 10 \end{bmatrix} \quad A21 = \begin{bmatrix} 5 & 1 \\ 45 & 9 \end{bmatrix} \quad A22 = \begin{bmatrix} 2 & 3 \\ 0 & -1 \end{bmatrix}$$

4. Perkalian Matriks - Algoritma

```
KaliMatriks2(in A,B: Matriks, inn : integer) : Matriks
Deklarasi
  i, j, k : integer
  A11, A12, A21, A22,
  B11, B12, B21, B22,
  C11, C12, C21, C22 : Matriks
Algoritma:
   if n = 1 then  return A x B   { perkalian biasa }
   else
      Bagi A  → A11, A12, A21, dan A22 ukurna n/2 x n/2
      Bagi B  → B11, B12, B21, dan B22 ukuran n/2 x n/2
      C11 <- KaliMatriks2(A11, B11, n/2) + KaliMatriks2(A12, B21, n/2)
      C12 <- KaliMatriks2(A11, B12, n/2) + KaliMatriks2(A12, B22, n/2)
      C21 <- KaliMatriks2(A21, B11, n/2) + KaliMatriks2(A22, B21, n/2)
      C22 <- KaliMatriks2(A21, B12, n/2) + KaliMatriks2(A22, B22, n/2)
      return C  { C adalah gabungan C11, C12, C13, C14  }
   endif
```

```
function Tambah(input A, B : Matriks, input n :
integer) :  Matriks
Deklarasi
  i, j, k : integer
Algoritma:
   for i <- 1 to n do
     for j <- 1 to n do
        C[i,j] <- A[i,j] + B[i,j]
     od
   od
   return C
```

Kompleksitas wakturnya :

$$T(n) = \begin{cases} a & ,n = 1 \\ 8T(n/2) + cn^2 & ,n > 1 \end{cases}$$

4. Perkalian Matriks - Kompleksitas

Kompleksitas waktu perkalian matriks seluruhnya adalah:

$$T(n) = \begin{cases} a & , n = 1 \\ 8T(n/2) + cn^2 & , n > 1 \end{cases}$$

yang bila diselesaikan, hasilnya adalah:

$$T(n) = O(n^3)$$

Hasil ini tidak memberi perbaikan kompleksitas dibandingkan dengan algoritma *brute force*.

Dapatkah kita membuat algoritma perkalian matriks yang lebih baik?

4. Algoritma Perkalian Matriks - Strassen

Hitung matriks antara:

M1 = (A12 – A22)(B21 + B22)

M2 = (A11 + A22)(B11 + B22)

M3 = (A11 – A21)(B11 + B12)

M4 = (A11 + A12)B22

M5 = A11 (B12 – B22)

M6 = A22 (B21 – B11)

M7 = (A21 + A22)B11

maka,

C11 = M1 + M2 – M4 + M6

C12 = M4 + M5

C21 = M6 + M7

C22 = M2 – M3 + M5 – M7

Kompleksitas waktu algoritma perkalian matriks Strassen:

$$T(n) = \begin{cases} a & , n = 1 \\ 7T(n/2) + cn^2 & , n > 1 \end{cases}$$

yang bila diselesaikan, hasilnya adalah

$$T(n) = O(n^{\log 7}) = O(n^{2.81})$$

Selesaikan T(n) sehingga diperoleh $O(n^{2.81})$

Latihan :

- **Selesai kompleksitas perkalian matrik metode Strasen O($n^{2,81}$)**